

Automatic Mesh Motion, Topological Changes and Innovative Mesh Setup for I.C.E. CFD Simulations

Tecniche Innovative per la Generazione, il Trattamento e il Moto di Mesh per la Simulazione di Motori a Combustione Interna con Codici di Calcolo CFD

T. Lucchini, G. D'Errico

Dipartimento di Energetica, Politecnico di Milano, Via La Masa 34, 20158 Milano.

Fax: ++39-2-23998566, e-mail: tommaso.lucchini@polimi.it

ABSTRACT

Mesh generation, handling and motion have been a bottleneck for 3-D simulation of in-cylinder phenomena of internal combustion engines. The geometry is complex and there are moving boundaries such as piston and valves. Due to the complex geometry, the mesh generation can be difficult and unstructured grids are generally used. Furthermore piston and valve motion require to move the mesh points and promote the need to change the mesh topology for different piston and valve positions, to keep a high mesh quality.

A new approach for moving meshes in internal combustion engine simulations is presented in this work. It combines the automatic mesh motion techniques with a series of topological changes which allow to add or remove cell layers or to deal with dynamically connected surfaces like valves or scavenging ports. The proposed approach should simplify the common methodology used for engine simulations where usually the point motion is provided from pre-calculated positions or by interaction with a pre-processor or a mesh generator.

A mesh setup for both two-stroke and four-stroke engines has been proposed and an incompressible flow solver was developed in order to test if the governing equations are correctly solved on moving meshes with topological changes.

INTRODUCTION

Nowadays simulation tools can provide very useful information for internal combustion engine design and optimization in a relatively short time. In particular, CFD codes give a better insight into the physical processes which take place within the cylinder and some other components of the intake and exhaust systems. Concerning in-cylinder simulations, the Finite Volume Method (FVM) is applied to moving and unstructured meshes, in order to model the behaviour of turbulent, compressible and reactive flows. This makes this subject very demanding and requires a reliable computational tool. On one side it has constantly to deal with new physical and numerical models, and on the other it must be flexible with respect to mesh structure and geometry handling to accommodate moving boundaries.

While the motion is solely defined on boundary points, most CFD codes require to specify the position of every vertex for any time-step in the mesh. This can be done in different ways, most commonly using "mesh generation" techniques, like smoothing. In practice this is quite limiting, as it becomes difficult to prescribe solution-dependent motion or perform mesh motion on dynamically adapting meshes.

In the present work an automatic mesh motion algorithm has been adopted. It uses a second-order FEM scheme, in which the user defines how the vertices on the boundary move, using a set of boundary conditions [1]. As nothing is done in advance of the solution, the boundary motion is "built into to code" and can be an arbitrary function of the solution. Then a Laplace equation is solved on the vertices to calculate the motion of all vertices based on the boundary motion. This equation is solved directly on vertices as doing it on volumes is not satisfactory. Additionally, to create a valid mesh, it must be guaranteed that no "cross-over" in the mesh is created when executing mesh motion. In this way the position of internal points is determined from the prescribed boundary motion so that the initial mesh remains valid.

To preserve the mesh quality during extreme boundary deformations due to piston and valve motion, the number of the cells in the mesh needs to be changed. For this reason a set of "topological changes" has been implemented, which concerns the possibility of attaching or detaching boundaries, adding or removing cell layers and sliding cells interfaces.

Finally, the proposed algorithms have been embedded in the object-oriented, open-source *OpenFOAM* code [2], and tested on mesh configurations aimed at reproducing the most common ones used in internal combustion engine simulations.

FINITE VOLUME METHOD ON MOVING MESHES

The integral form of the conservation equation for a tensorial property ϕ defined per unit mass in an arbitrary moving volume V bounded by a closed surface S states [1]:

$$\frac{d}{dt} \int_V \rho \phi dV + \oint_S d\mathbf{s} \cdot \rho (\mathbf{u} - \mathbf{u}_b) = - \oint_S d\mathbf{s} \cdot \rho \mathbf{q}_\phi + \int_V \mathbf{S}_\phi dV \quad (1)$$

where ρ is the density, \mathbf{u} is the fluid velocity, \mathbf{u}_b is the boundary velocity and \mathbf{q}_ϕ and \mathbf{S}_ϕ are the surface and volumes sources/sinks of ϕ , respectively. As the volume V is no longer fixed in space, its motion is captured by the motion of its bounding surface S by the velocity \mathbf{u}_b .

Compared with the FVM on a static mesh (e.g. [4]), the second-order FV discretization of the Equation (1) shows only two differences: the temporal derivative introduces the rate of change of the cell volume and the mesh motion flux accounts for the grid convection. The relationship between the two is governed by the space conservation law [5]:

$$\frac{d}{dt} \int_V dV - \oint_S d\mathbf{s} \cdot \mathbf{u}_b = 0 \quad (2)$$

The unstructured FVM method splits the computational space into a finite number of polyhedral cells bounded by complex polygons which do not overlap and completely cover the domain. The temporal dimension is split into a finite number of time-steps and the equations are solved in a time-marching manner. A sample cell around the computational point P located in its centroid, a face f , its area vector \mathbf{S}_f and the neighbouring computational point N are shown in Figure 1.

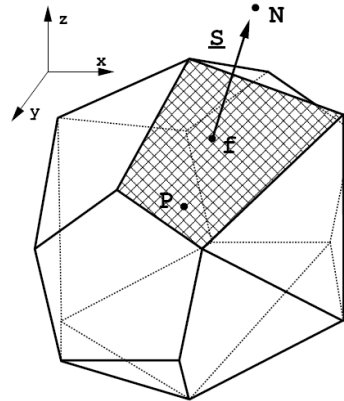


Figure 1 - Finite volume cell.

While Equation (2) is always satisfied in the integral form, it also needs to be preserved in the discrete form:

$$\frac{V_P^n - V_P^0}{\Delta t} - \sum_f F_s = 0 \quad (3)$$

For this reason, the mesh motion flux \mathbf{F}_s is calculated as the volume swept by the face f in motion during the current time-step rather than from the grid velocity \mathbf{u}_b , making it consistent with the cell volume calculation [1].

MESH DEFINITION AND MOTION

It is necessary to distinguish from *boundary motion* and *internal point motion*. In internal combustion engines boundary motion is prescribed by the engine geometry and speed.

Internal point motion has to accommodate changes in the domain shape and preserve the validity and quality of the mesh. Furthermore, it influences the solution only through mesh-induced discretisation errors [6] and is therefore detached from the remainder of the problem. Consequently, internal point motion can be specified in a number of ways, ideally without user interaction [1].

Automatic mesh motion provides great advantage, both in terms of easier and faster case setup, also allowing for dynamically changing topology or automatic improvements of the mesh quality. The objective of

automatic mesh motion is to determine internal point motion (not involving topological changes) to conform with given boundary motion while preserving mesh validity and quality.

Mesh definition and validity. A valid mesh is a pre-requisite for a good numerical solution and a critical ingredient of automatic mesh motion. The investigation of mesh validity can be separated into *topological* and *geometrical* tests. The first group contains the tests that can be performed without knowing the actual point positions, while the second deals with the shape of cells and the boundary [1].

Face based mesh definition. In the *face-addressed* mesh definition, a polyhedral mesh for the FVM is defined by the following components:

- A list of points. For every point, its space co-ordinates are given; the point label is implied from its location in the list.
- Every point must be used in at least one face;
- A list of polygonal faces, where a face is defined as an ordered list of point labels. Faces can be separated into internal (between two cells) and boundary faces. Every face must be used by at least one cell;
- A list of cells defined in terms of face labels. Note that the cell shape is unknown and irrelevant for discretisation;
- Boundary faces are grouped into patches, according to the boundary condition. A patch is defined as a list of boundary face labels.

Face orientation is determined using the right-hand rule and the face list will first collect all internal faces and then all boundary faces patch by patch in the order of patch definition. Internal faces are ordered to contain all faces from the first cell with the increasing neighbour label, followed by the faces owned by the second cell *etc.* This approach has proven to be robust and easy to handle as it enforces strict and unique face ordering [1].

Topological tests. Topological validity tests consist of the following criteria:

- A point can appear in a face only once;
- A face can appear in a cell only once. A face cannot belong to more than two cells. A boundary face can belong to only one patch;
- Two cells can share no more than one face;
- Collecting all faces from one cell and decomposing faces into edges, every edge must appear in exactly two cell faces;
- Collecting all faces from the boundary and decomposing faces into edges, every edge must appear in exactly two boundary faces.

The first four conditions control the validity of the mesh definition while the last two check that all cells and the boundary hull are topological closed. Additionally, mesh ordering rules are checked and enforced.

Geometrical tests. Geometrical tests deal with the positivity of faces areas and cell volumes, as well as convexity and orientation requirements. Geometrical validity criteria can be summarized as follows:

- All faces and cells must be weakly convex;
- All cells must be geometrically closed: the sum of the outward-pointing face area vectors for a cell faces must be zero to machine tolerance;
- The boundary must be geometrically closed;
- For all the internal faces, the dot-product of the face area vectors \mathbf{s}_f and the $\mathbf{d}_f = \overline{PN}$, Figure 1, must be positive; this is usually termed the *orthogonality test*:

$$\mathbf{d}_f \cdot \mathbf{s}_f > 0$$

We shall assume the existence of a topologically and geometrically valid mesh as a starting point for automatic mesh motion [1]. During mesh motion, mesh topology remains unaffected and only the point positions change. Thus, preserving the mesh quality only relates to the geometrical tests. Moreover, once the convexness and orthogonality tests are satisfied, an initially valid mesh remains valid if no triangles or tetrahedra are inverted [1].

POLYHEDRAL VERTEX-BASED MOTION SOLVER

The polyhedral mesh motion solver proposed by Jasak and Tukovic [1, 3] takes lead from the Finite Element Method (FEM) practice, decomposing the mesh into tetrahedrals. To provide vertex motion the Laplacian equation is solved in the FEM mesh.

Cell decomposition. Every polyhedral cell is split into tetrahedra by splitting its faces into triangles and introducing a point in cell centroid. Consistency in tetrahedral connectivity is obtained by using identical face decomposition for both cells sharing an internal face [1, 3].

Cell-and-face decomposition has been used. It introduces an additional point in all cell centres and a point in all cell faces, as shown in Figure 2.

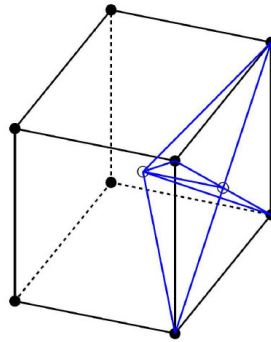


Figure 2 - *Cell-and-face decomposition of a polyhedral cell into tetrahedra.*

Motion Equation. The Laplace operator:

$$\nabla \cdot (\gamma \nabla \mathbf{u}_p) = 0 \quad (4)$$

with constant or variable diffusion field γ is chosen to govern mesh motion. Here, \mathbf{u}_p is the point velocity field used to modify point positions:

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \mathbf{u}_p \Delta t \quad (5)$$

The choice to solve for motion velocity is related to the fact that it changes slower than point position and thus a better initial guess is available. For stationary meshes the velocity solution equals to zero everywhere and is less polluted by round-off errors than the (linear) point position field. For constant-velocity deformation the cost of solving the motion equation in terms of velocity becomes trivial (a good initial guess is available), which is not the case if point position is chosen as the primitive variable. For better precision, the motion velocity on the boundary is calculated from the current and desired point position and the time-step. This approach avoids the accumulation of round-off errors associated with solving for motion velocity and using point position [1, 3] .

Boundary conditions for the motion equation are enforced from the known boundary motion; this may include free boundaries, symmetry planes, prescribed motion boundary *etc.*

TOPOLOGICAL CHANGES

In internal combustion engines, the valves and piston motion change radically the cylinder volume and shape. During compression, for example, the cylinder volume can be reduced by a factor of ~ 15 . Furthermore, when valves are open, cylinder is not a closed volume any more, but it exchanges fluid with the intake and exhaust ports.

The mesh motion can be correctly described by the automatic mesh motion solver, but in extreme cases of boundary deformations, mesh quality can be only preserved by changing the topology of the mesh. A *topological change* is any mesh operation which changes the connectivity of the mesh or the number of points, faces or cells in it. Topological changes typically used in IC engine simulations are attach/detach boundaries, cell layer addition/removal and sliding mesh interfaces [7].

Mesh operations requiring topological changes will be collectively named *mesh modifiers*. After any topological change the mesh has to remain valid and to satisfy all the tests previously described. The mesh modifiers used for engine simulations are now described.

Cell layer addition/removal. The layer *addition/removal* mesh modifier is specified by a set of oriented faces defining the base layer and the thickness threshold for layer addition and removal. Figure 3 shows layer addition and removal during the motion of a cone in a rectangular mesh [8].

When the mean cell thickness exceeds a prescribed value, new cells are added. On the contrary, when the thickness is lower than a prescribed minimum thickness, cells are removed.

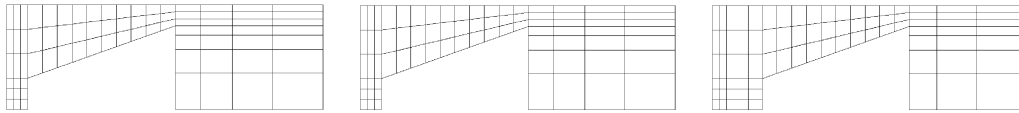


Figure 3 - A moving cone in a rectangular mesh adds and removes cell layers during its motion.

Sliding interface. This mesh modifier allows to deal with sliding mesh interfaces, by changing dynamically the cell connectivity, in order to keep a high mesh quality in complex geometries like internal combustion engines or mixers. In Figure 4, the sliding interface mesh modifier is applied to a mixer geometry.

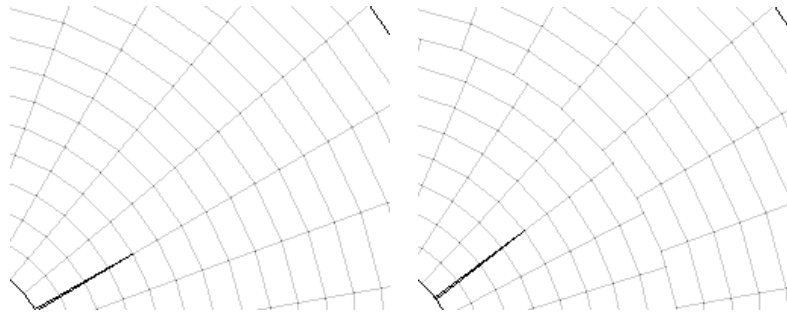


Figure 4 - Sliding mesh interface applied to the interior part of a mixer mesh.

Attach/detach boundary. Another issue which does not directly involve mesh motion is related to the volume connectivity. In internal combustion engines, when a valve is closed, it perfectly separates the cylinder volume from the intake or exhaust ports. In the past years the problem of simulating the contact between two surfaces was solved by leaving an "inert" layer of cells between the two surfaces. To ensure no fluid dispersion, zero velocity is imposed to these cells. Hard coding is necessary for this solution and it is basically wrong to involve the fluid motion to solve a problem which is strictly related with the mesh topology.

The mesh modifier *attach-detach boundary* represents an alternative approach. It creates two new boundary entities from a list of internal faces, so as to separate or connect two distinct volumes in the same mesh. In Figure 5 the use of attach-detach boundary is shown when a T junction is attached and detached [8].

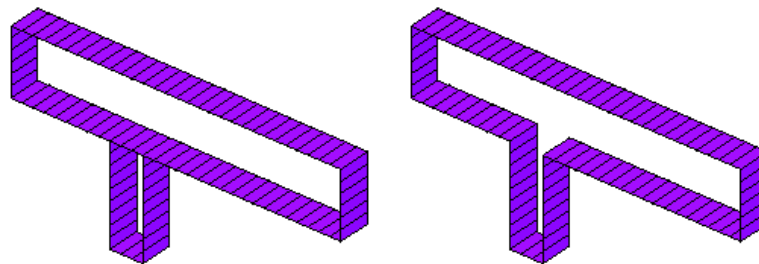


Figure 5 - Attach-detach boundary mesh modifier applied to a T-junction.

Application of the finite volume method. Equation (1) represents the FVM method applied on a moving mesh. When the mesh is also changing its topology the number of faces and cells varies time step by time step. Scalar and vector fields are mapped from the old mesh to the new mesh and surface fluxes are recalculated when sliding interfaces are used. Finally, new boundary conditions are imposed when boundary are attached.

AUTOMATIC MESH MOTION FOR INTERNAL COMBUSTION ENGINES

Automatic mesh motion and topological changes have to be combined in an internal combustion engine simulation. The mesh motion algorithm works as follows [9,tlthe]:

1. All the sliding interfaces and boundaries are detached. In this way all the volumes in the mesh are geometrically and topologically separated.
2. The automatic mesh motion is performed in all the volumes. Thus the mesh motion is independent for any distinct geometrical entity in the mesh. Layers of cells are added or removed in this phase.

3. The sliding interfaces are re-attached and, if necessary, boundaries are detached. In this way the connectivity of the mesh is re-established.

MESH SETUP FOR TWO-STROKE ENGINES

In two stroke engines it is necessary to account for the piston motion and the presence of the scavenging ports. To keep the mesh aspect ratio close to the optimum value, the piston should add or remove layers of cells during its motion, but when the piston approaches TDC, the possibility to deform mesh should have to be taken into account so as to refine the mesh during the main part of the combustion phase.

The fluid-exchange process takes place when the piston uncovers the scavenging ports. To account for this, the sliding interface mesh modifier dynamically connects the liner and the ports surfaces.

Figure 6 shows the mesh setup for a two-stroke engine geometry. The liner (*light blue*) is dynamically connected with the scavenging ports (*yellow*) through a sliding interface [9,10].

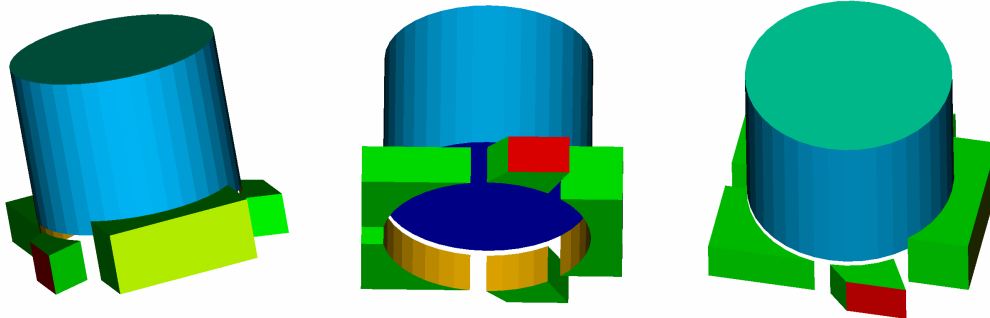


Figure 6 - Two stroke engine mesh setup. Piston (*blue*), intake ports (*red*), liner (*light blue*), cylinder head (*dark green*), scavenging ports (*yellow*), exhaust ports (*light green*).

Mesh Motion. Additional input data are required to perform mesh motion. They concerns the engine geometry (bore, stroke and rod length), and the value of diffusion for the Laplace equation of motion which is solved accounting for the piston boundary motion.

Details of the mesh motion are shown in Figure 7, where it is possible to distinguish layering and deformation modes.

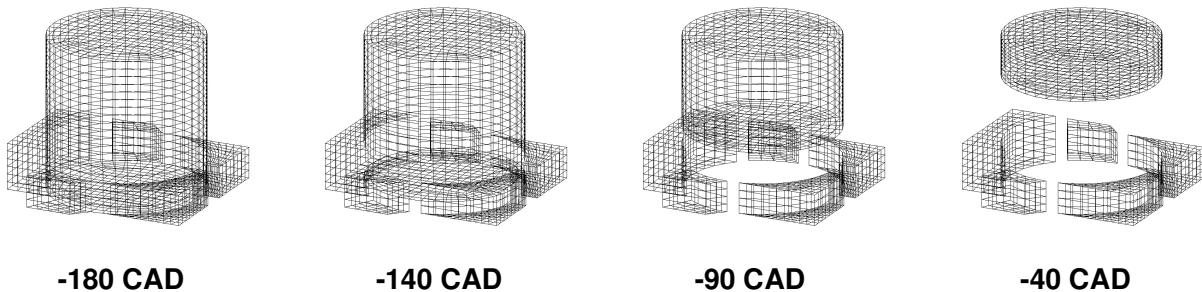


Figure 7 - Details of the grid motion for the two-stroke engine mesh.

The dynamic connection between the scavenging ports and the liner can be seen in Figure 8, before the scavenging process begins, the scavenging ports are fully closed. When the piston goes down, they are uncovered until they are fully opened [9,10].

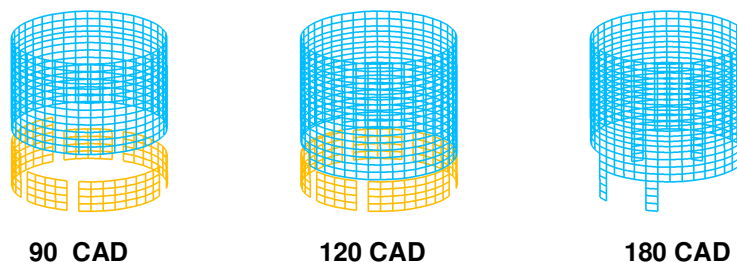


Figure 8 - Dynamic connection between the liner and the scavenging ports surfaces.

MESH SETUP FOR FOUR-STROKE ENGINES

A four stroke engine with N valves has $2N + 1$ moving boundaries. The piston and also the top and the bottom of the valves are moving. All the moving boundaries should be allowed to add or remove layers and the mesh quality should be always preserved without distortions. To account for this, a series of sliding interfaces has to be used, and for this reason it is necessary to split the cylinder volume in several distinct meshes during the mesh generation process as follows:

- Volumes displaced by the valves (*valve volume*).
- Cylinder volume

Any *valve volume* will be coupled with the cylinder mesh through a sliding interface. Thus the cylinder will require N additional boundary patches. The cylinder and valve meshes are shown in Figures 9-10.

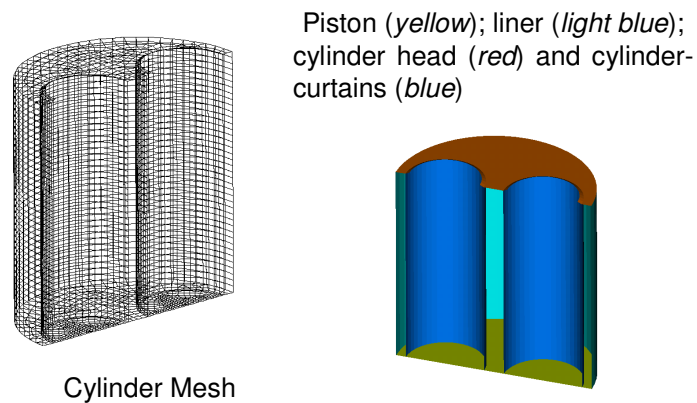


Figure 9 - Details of the cylinder mesh and its surfaces.

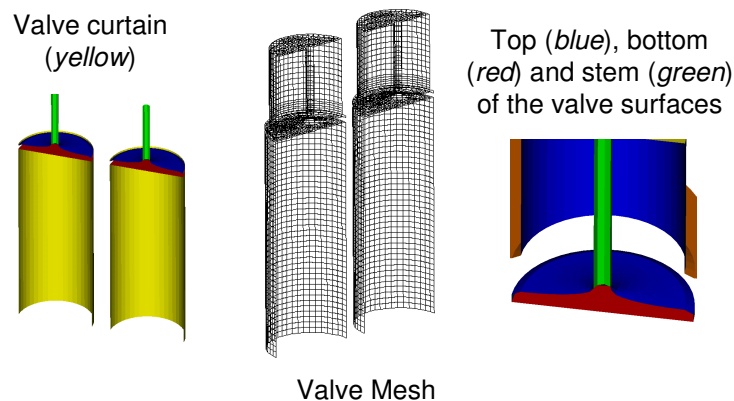


Figure 10 - Details of the valve mesh and its surfaces.

The valve motion implies topological changes. Top and bottom of the valves add and remove layers during their motion, while a surface called *valve curtain* is used to couple the valve volumes with the rest of the mesh. Each valve surface requires its own patch. Figure 10 shows the detail of the top, bottom, stem of the valve, and also the valve curtain surface.

Cylinder mesh has a series of "holes" which will be used to seat the valve meshes, and it can be seen in Figure 9. Any "hole" surface will be named as *cylinder curtain*. The piston, liner and cylinder head surfaces can be identified as well as the *cylinder curtain* surfaces used to connect the valve meshes with the cylinder.

The complete mesh is presented in Figure 11. Valves and cylinder meshes are distinct and they will be joined by a series of sliding interfaces. Since the main aims of the work were to test the mesh motion solver and the topological changes, the geometry is simplified: the intake and exhaust ports are vertical and there is no piston bowl. However, the proposed algorithm is working also on more complex engine geometries.

The valve closure is simulated by using the attach-detach mesh modifier. Once the mesh is generated, a list of internal faces has to be specified. These internal faces are used by the mesh modifier to detach the cylinder mesh from the port mesh when the valve is closed [9,10].

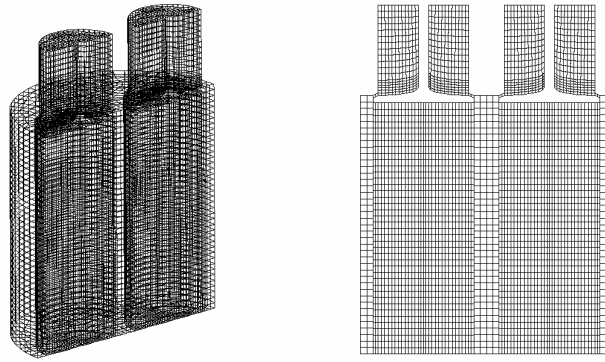


Figure 11 - Full engine mesh configuration (left) and a section of the cutting plane (right).

Automatic mesh motion is performed, accounting both for piston and valve boundary motion. As for the two-stroke engine mesh, it is possible to switch from layering to deformation mode. In Figure 12, the mesh motion and topological changes for the proposed four stroke engine layout are shown.

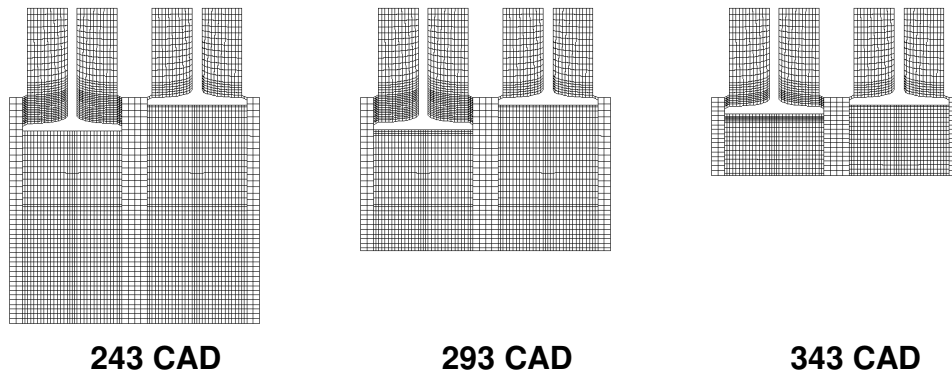


Figure 12 - Mesh motion for the four-stroke engine.

RESULTS

To test the finite volume method on moving meshes with topological changes, an incompressible solver has been developed. The choice of an incompressible solver was made mainly to reduce the number of equations to be solved and not to deal, in this preliminary stage with more complex problems like transonic flows and combustion.

Since the fluid is incompressible only the continuity and the momentum equations are solved. The PISO algorithm is used to compute the pressure field. Since the fluid is not compressible, only the gas exchange process can be simulated, while it is not possible to account for the compression and expansion phases. It must be pointed out that the gas exchange process is the most interesting phase, on the topological point of view, because all the mesh boundaries are moving.

The tested mesh consists of ~ 45000 cells at BDC, which are reduced to ~ 20000 when the piston is at TDC. The computational mesh is the same displayed in Figure 10-11. Two transport equations are solved for the pressure and velocity field, considering water as the working fluid. The boundary conditions account for the piston boundary motion and the amount of total pressure imposed at the intake and exhaust ports.

The main engine data are summarized in Table 1.

Bore	120 mm
Stroke	75 mm
Rod Length	147 mm
Engine Speed	1500 rpm
Maximum Valve Lift	11.5 mm
EVO	160 ATDC
EVC	380 ATDC
IVO	340 ATDC
IVC	560 ATDC

Table 1 - Main engine data.

Figure 13 shows details of the exhaust and the intake strokes on two cutting planes passing through the valve axis and orthogonal to the symmetry plane. The surfaces are coloured like pressure and velocity vectors are also displayed. The qualitative behaviour of the pressure and velocity is satisfying. There is a pressure drop through the valve section where the fluid accelerates [9].

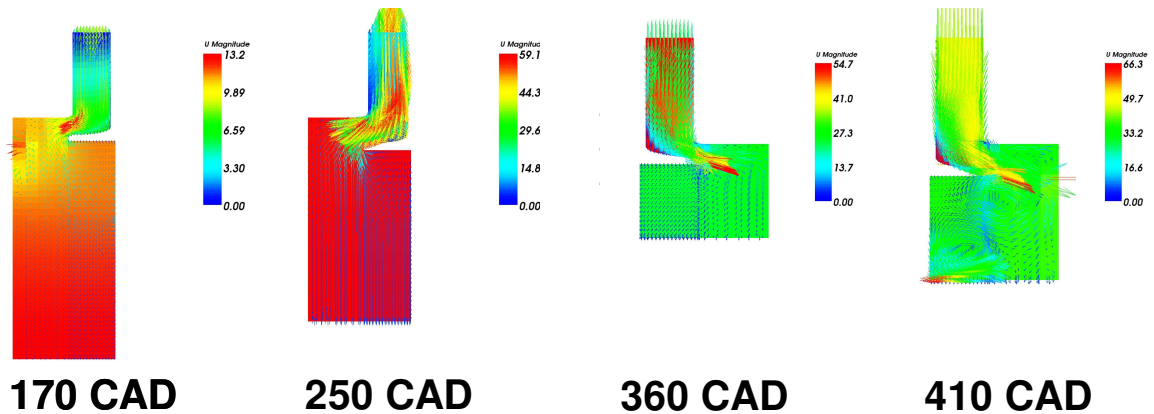


Figure 13 - Velocity vectors and pressure distribution during the exhaust and intake strokes.

To verify that equations are correctly solved, since the fluid is incompressible, the volumetric flow imposed by the piston should be equal to the sum of the volumetric flows on the intake and exhaust ports. Figure 14 shows that the mass conservation check is perfectly respected, thus the equations are correctly solved [10].

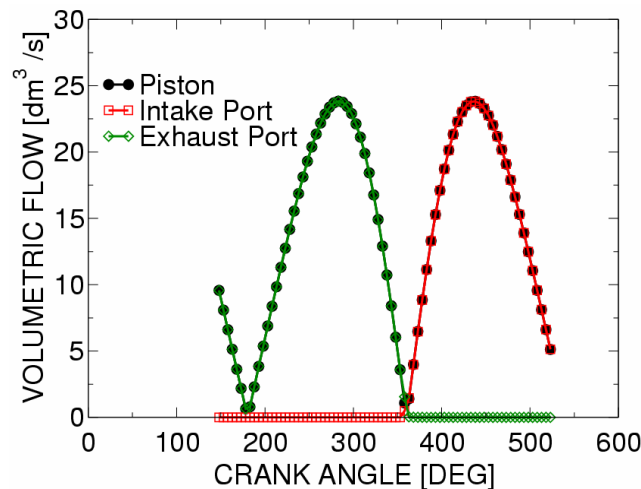


Figure 14 - Mass conservation check for the engine case during the fluid-exchange phase.

CONCLUSIONS

A new approach to account for mesh motion in internal combustion engine simulations has been presented in the present work. The vertex motion is provided by solving the Laplace equation of motion in a tetrahedral decomposition of the mesh, in this way the mesh motion is solution-dependent and does not require any pre-processing.

To preserve the mesh quality during the simulation, the mesh topology is changed by adding or removing cell layers and using sliding interfaces to connect dynamically the parts of the mesh which slides each other. The valve closure phase is simulated by creating two distinct boundary entities from a list of internal faces, and this allows to split the mesh in separated parts without forcing any condition to the fluid flow.

A mesh setup for both two- and four-stroke engines has been proposed. The mesh generation process has to account for the presence of sliding interfaces, while only one additional pre-processing operation is required, to specify the list of internal faces to be detached. However the proposed mesh setup is relatively user-friendly.

The four-stroke engine mesh has been tested with an incompressible solver to verify the performance of the Finite Volume Method when it is applied to moving meshes with topological changes. This requires fields to be remapped and surface fluxes to be recalculated. The results obtained in terms of mass conservation ensure that the proposed work can be easily extended for the simulation of the compressible flow in real internal combustion engines.

ACKNOWLEDGEMENTS

The authors are grateful to dr. Hrvoje Jasak for his support in the development of this work.

REFERENCES

- [1] H. JASAK and Z. TUKOVIC. *Automatic Mesh Motion for the Unstructured Finite Volume Method*. Journal of Computational Physics, (To be Published), 2004.
- [2] OpenFOAM web-site: <http://www.openfoam.org>, 2006
- [3] Z. TUKOVIC. *Finite Volume Method in Domains of Varying Shape*. PhD thesis, University of Zagreb, Croatia, 2005.
- [4] H. JASAK and A. GOSMAN. *Automatic Resolution Control for the Finite Volume Method. Part 1: A-posteriori Error Estimates*, Numerical Heat Transfer. Numerical Heat Transfer, Part B, Vol. 38, pages 237-256, 2000.
- [5] I. DEMIRDZIC and M. PERIC. *Space Conservation Law in Finite Volume Calculations of Fluid Flow*. Int. J. Num. Meth. Fluids, Vol. 8, pages 1037-1050, 1998.
- [6] H. JASAK. *Error Analysis and estimation for the finite volume method with applications to fluid flows*. PhD thesis, Imperial College of Science, Technology and Medicine, London, 1996.
- [7] H. JASAK, H.G. WELLER, and N. NORDIN. *In-Cylinder CFD Simulation Using a C++ Object-Oriented Toolkit*. SAE Paper, 2004-01-0110, 2004.
- [8] H. JASAK. FOAM CFD web-site: <http://www.foamcfd.org>, 2006.
- [9] T. LUCCHINI. *Internal Combustion Engine Simulation in OpenFOAM*. First OpenFOAM workshop, Zagreb, 2006.
- [10] T. LUCCHINI. *Prediction of Combustion and Pollutant Emissions in Internal Combustion Engines*, PhD thesis, Politecnico di Milano, 2006.